ダウンロード・データあります

組み込みRustのライブラリ 便利クレート探偵団

第7回 ビット・フィールドへのアクセスを簡単にする bit_field と bitfield

中林 智之

Rustは組み込みで使えるプログラミング言語として注目されています．そんなRustの組み込み開発で役立つライブラリ（クレート）を本連載では紹介します．

今回から数回にわたりRustでビット・レベルの操作を行うのに便利なクレートを紹介します．

今回はビット・フィールドへのアクセスを簡単にするbit_field[1]とbitfield[2]を紹介します．

ビットおよびビット・フィールドにアクセスするシンプルなインターフェースを提供するのがbit_fieldでC言語のビット・フィールドに近い機能を提供するのがbitfieldとなります．

Rustには直接的なビット・フィールドの機能がない

C言語ではビット・レベルの操作を行う機能としておなじみのビット・フィールドが存在しています．

C言語のビット・フィールドは，1バイト未満の整数値をひとまとめにしてメモリ領域を節約する機能です．構造体を宣言するときにビット単位のフィールド幅を指定します．

リスト1の例でbyte_tは，1ビットのflag1，flag2と6ビットのfield1を含む1バイトの構造体として宣言されています．ビット単位で宣言したフィールドも通常の構造体のフィールドと同じようにbyte.flag1としてドットでアクセスします．

Rustには2024年11月時点では，このようなビット・フィールドの機能が言語の機能としては存在していません．そのため，Rustでビット・レベルの操作をする場合は，いずれかの方法を取ることになります．

1，ビット操作を手で書く
2，目的に応じたクレートを使う

少しならビット操作を手で書くのも悪くないですが，できれば手軽にビット操作をしたいところです．

そこで，目的に応じてビット操作を便利にできるクレートを使います．それが今回紹介するbit_fieldとbitfieldクレートです．

リスト1　C言語でビット・フィールドを扱う場合

```
#include <assert.h>
#include <stdint.h>

struct byte_t {
    uint8_t flag1 : 1;
    uint8_t flag2 : 1;
    uint8_t field1 : 6;
};

int main(void)
{
    struct byte_t byte = { 1, 0, 42 };
    assert(sizeof(byte) == 1);
    assert(byte.flag1 == 1);
    assert(byte.flag2 == 0);
    assert(byte.field1 == 42);

    return 0;
}
```

ビット単位のアクセスを容易にするbit_fieldクレート

bit_fieldはRustの整数型に対して，各ビットまたはビット・フィールドにアクセスするBitFieldトレイトを提供します．

また，整数型の配列（スライス）をまとめて扱えるBitArrayトレイトもあります．それぞれ順番に見てみましょう．

● BitFieldトレイトで整数型のビット・フィールドを操作する

Cargo.tomlにbit_fieldクレートの依存を追加します．

```
[dependencies]
bit_field = "0.10.1"
```

まずはu8のデータに1ビットの操作（getとset）をするプログラムを見てみましょう（**リスト2**）．

データの特定ビットを操作する場合はget_bit()とset_bit()を使います．値はboolean（trueまたはfalse）として扱います．

ビット・フィールドを操作する場合はget_bits()とset_bits()を使います．操作するビット・フィー

ダウンロード・データはこちら
https://interface.cqpub.co.jp/2501crate/